Computational Statistics
Lab 1

Group 03
Mohsen Pirmoradiyan, Ahmed Alhasan, Yash Pawar

January 26, 2020

Contents

Question 1: Be careful when comparing

```
x1 <- 1/3; x2 <- 1/4
if (x1 - x2 == 1/12) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}
```

```
## [1] "Substraction_is_wrong"
```

```
x1 <- 1; x2 <- 1/2
if (x1 - x2 == 1/2) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}
```

```
## [1] "Substraction_is_Correct"
```

- In the first case x1 = 1/3 can not be equal to it's mathematical representation (for base 10 the fraction continues to infinity), therefore it have to be rounded to the maximum number of digits allowed.

```
options(digits = 22)
x1 <- 1/3; x2 <- 1/4
x1
```

```
## [1] 0.33333333333333331
```

```
x2
```

```
## [1] 0.25
```

- In the second case the result is correct because both numbers equal to their mathematical value (only integers within the allowable range and fractions whose denominators are factors of base 10 i.e. 1/2 and 1/5 can have exact represenation of their mathematical value).

```
x1 <- 1; x2 <- 1/2
x1
```

```
## [1] 1
```

```
x2
```

```
## [1] 0.5
```

- To solve this problem we can either use all.equal() function to test near equality.

```r
x1 <- 1/3; x2 <- 1/4
if (isTRUE(all.equal(x1 - x2, 1/12))) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}
```

```
## [1] "Substraction_is_Correct"
```

- Or we can round both numbers before testing equality.

```r
x1 <- 1/3; x2 <- 1/4
if (round((x1 - x2), 4) == round((1/12),4)) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}
```

```
## [1] "Substraction_is_Correct"
```

Question 2: Derivative

```r
f <- function(x) {x}
df <- function(x){
  eps <- 10^-15
  (f(x + eps) - f(x)) / eps
}
```

```r
df(1)
```

```
## [1] 1.1102230246251565
```

```r
df(10000)
```

```
## [1] 0
```

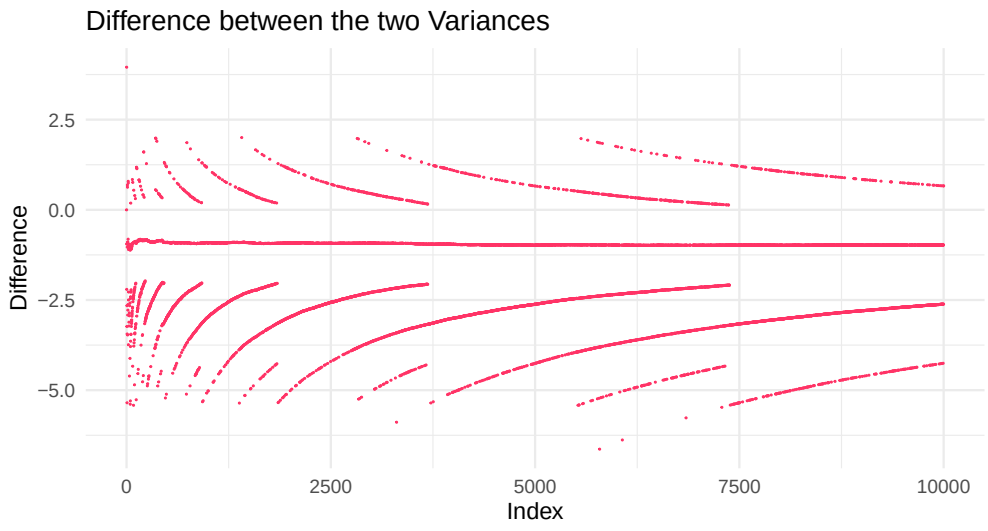
- The true value of both derivatives is equal to 1, however when x = 10000 and epsilon is very small, 10000 + 10^-15 will still be 10000 (the small value of epsilon is neglected "underflow"), therefore the substraction in the nominator becomes 0 and the derivative will yield 0.

Question 3: Variance

```
myvar <- function(x_vec) {
  n <- length(x_vec)
  (1/(1-n)) * (sum(x_vec^2) - (1/n) * sum(x_vec)^2)
}

x_vec <- rnorm(10000, mean = 10^8, sd = 1)

library(ggplot2)
Y1 <- function(x){
  n <- length(x)
  y <- numeric(length = n)
  for(i in 2:n) {
    X    <- x_vec[1:i]
    y[i] <- myvar(X) - var(X)
  }
  #plot(y, cex = 0.5)
  df <- data.frame(c(1:n),y)
  ggplot(df) +
    geom_point(aes(x = df[,1], y = df[,2]),
               size = 0.01,
               color = "#FF3366") +
    labs(title = "Difference between the two Variances",
         x = "Index",
         y = "Difference")+
    theme_minimal()
}
Y1(x_vec)
```

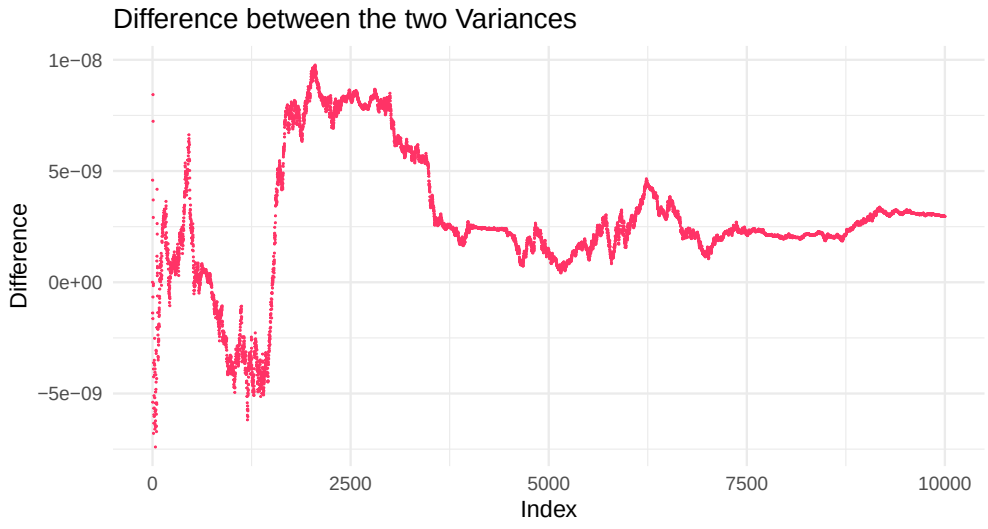


- myvar() function behave wildly because of what is called Catastrophic Cancelation, where the resulted value of sum(x_vec^2) and (1/n) * sum(x_vec)^2 can not be accurately represented by the machine because of the limited storage, so in this case they are rounded to the same number and canceled each other despite them being actually different.

```r
var_YC <- function(v_x){
  ## v_x is a numerical vector of length greater than 2
  ## this function calculates the sample variance
  ## using the Youngs and Cramer algorithm
  T   <- v_x[1]
  RSS <- 0
  n   <- length(v_x)
  for (j in 2:n){
    T   <- T + v_x[j]
    RSS <- RSS + ((j * v_x[j] - T)^2) / (j*(j-1))
  }
  RSS /(n-1)
}

Y2 <- function(x){
  n <- length(x)
  y <- numeric(length = n)
  for(i in 2:n) {
    X    <- x_vec[1:i]
    y[i] <- var_YC(X) - var(X)
  }

  #plot(y, cex = 0.5)
  df <- data.frame(c(1:n),y)
  options(digits = 3)
  ggplot(df) +
    geom_point(aes(x = df[,1], y = df[,2]),
               size = 0.01,
               color = "#FF3366") +
    labs(title = "Difference between the two Variances",
         x = "Index",
         y = "Difference")+
    theme_minimal()
}
Y2(x_vec)
```



- In the Young-Cramer method, it avoid substracting two numbers of almost the same magnitude, so we get a close approximation to the variance function.

Question 4: Linear Algebra

```
options(digits = 5)
tecator <- readxl::read_xls("tecator.xls")

X <- as.matrix(tecator[,-c(1,103)])
y <- as.matrix(tecator[,103])

A <- t(X) %*% X
b <- t(X) %*% y

solve.default(A,b)
```

```
## Error in solve.default(A, b): system is computationally singular: reciprocal condition number = 7.13
```

- Because the variables are highly correlated with each other in the original data, so is matrix A which make it a singular matrix that is not invertible, therefore it is not possible to solve for coeffients.

```
kappa(A)
```

```
## [1] 1.1578e+15
```

- The condition number is very high that could imply the matrix is ill-conditioned to solve the problem

```
X_scaled <- scale(X)
y_scaled <- scale(y)

A_new <- t(X_scaled) %*% X_scaled
b_new <- t(X_scaled) %*% y_scaled


kappa(A_new)
```

```
## [1] 4.9047e+11
```

- When we scale the data the round-off error becomes less significant, even though the new condition number is lower it is not necessarily well-conditined, but we have lesser perturbation to deal with.

Appendix

```
#Question 1
x1 <- 1/3; x2 <- 1/4
if (x1 - x2 == 1/12) {
  print("Substraction_is_Correct")
```

```r
}else {
  print("Substraction_is_wrong")
}

x1 <- 1; x2 <- 1/2
if (x1 - x2 == 1/2) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}

options(digits = 22)
x1 <- 1/3; x2 <- 1/4
x1

x1 <- 1; x2 <- 1/2
x1
x2

x1 <- 1/3; x2 <- 1/4
if (isTRUE(all.equal(x1 - x2, 1/12))) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}

x1 <- 1/3; x2 <- 1/4
if (round((x1 - x2), 4) == round((1/12),4)) {
  print("Substraction_is_Correct")
}else {
  print("Substraction_is_wrong")
}


#Question 2
f <- function(x) {x}
df <- function(x){
  eps <- 10^-15
  (f(x + eps) - f(x)) / eps
}

df(1)
df(10000)


#Question 3
myvar <- function(x_vec) {
  n <- length(x_vec)
  (1/(1-n)) * (sum(x_vec^2) - (1/n) * sum(x_vec)^2)
}

x_vec <- rnorm(10000, mean = 10^8, sd = 1)
```

```r
library(ggplot2)
Y1 <- function(x){
  n <- length(x)
  y <- numeric(length = n)
  for(i in 2:n) {
    X    <- x_vec[1:i]
    y[i] <- myvar(X) - var(X)
  }
  #plot(y, cex = 0.5)
  df <- data.frame(c(1:n),y)
  ggplot(df) +
    geom_point(aes(x = df[,1], y = df[,2]),
               size = 0.01,
               color = "#FF3366") +
    labs(title = "Difference between the two Variances",
         x = "Index",
         y = "Difference")+
    theme_minimal()
}
Y1(x_vec)

var_YC <- function(v_x){
  ## v_x is a numerical vector of length greater than 2
  ## this function calculates the sample variance
  ## using the Youngs and Cramer algorithm
  T   <- v_x[1]
  RSS <- 0
  n   <- length(v_x)
  for (j in 2:n){
    T   <- T + v_x[j]
    RSS <- RSS + ((j * v_x[j] - T)^2) / (j*(j-1))
  }
  RSS /(n-1)
}

Y2 <- function(x){
  n <- length(x)
  y <- numeric(length = n)
  for(i in 2:n) {
    X    <- x_vec[1:i]
    y[i] <- var_YC(X) - var(X)
  }

  #plot(y, cex = 0.5)
  df <- data.frame(c(1:n),y)
  options(digits = 3)
  ggplot(df) +
    geom_point(aes(x = df[,1], y = df[,2]),
               size = 0.01,
               color = "#FF3366") +
    labs(title = "Difference between the two Variances",
         x = "Index",
         y = "Difference")+
```

```
    theme_minimal()
}
Y2(x_vec)

#Question 4
options(digits = 5)
tecator <- readxl::read_xls("tecator.xls")

X <- as.matrix(tecator[,-c(1,103)])
y <- as.matrix(tecator[,103])

A <- t(X) %*% X
b <- t(X) %*% y

solve.default(A,b)

kappa(A)

X_scaled <- scale(X)
y_scaled <- scale(y)

A_new <- t(X_scaled) %*% X_scaled
b_new <- t(X_scaled) %*% y_scaled


kappa(A_new)
```